RELATÓRIO

OTHELLO

PROGRAMAÇÃO I

Trabalho Prático 2021/2022

ALUNOS:
51972 – Miguel Costa
52183 – Rodrigo Marques

DOCENTES:
Prof. Dr. Nuno Lourenço
Prof. Dra. Teresa Gonçalves

Índice

Introdução

Como trabalho prático final para a cadeira de Programação I, foi-nos proposto a realização do jogo de tabuleiro chamado "Othello" (ou Reversi em inglês) no computador em linguagem C.

Othello é um jogo de tabuleiro 8x8 de estratégia para dois jogadores onde um tem peças pretas e as outras peças brancas. O objetivo dos jogadores é ter a maioria das peças da sua cor no final do jogo e para isso têm de virar o máximo de peças possíveis colocadas pelo seu adversário de acordo com as regras do jogo.

O objetivo principal com este projeto é adaptar as regras do jogo em código imperativo processual C através de funções para tornar possível jogar o jogo no terminal contra o computador. Enquanto o objetivo do relatório é sintetizar e demonstrar o nosso método de raciocínio ao longo do código, enquanto explicamos o funcionamento de cada função.

Desenho da aplicação

Ao iniciar o projeto, deparamo-nos com muita informação e tarefas a fazer para o jogo funcionar de acordo com as regras previstas. Então, para ajudar na nossa orientação do trabalho, decidimos, em certos momentos, escrever as nossas ideias e os passos que teríamos eventualmente de tomar para o desenvolvimento do projeto.

Abaixo, encontram-se imagens das ideias desenvolvidas ao longo do trabalho, tal como a descrição de cada uma das imagens:

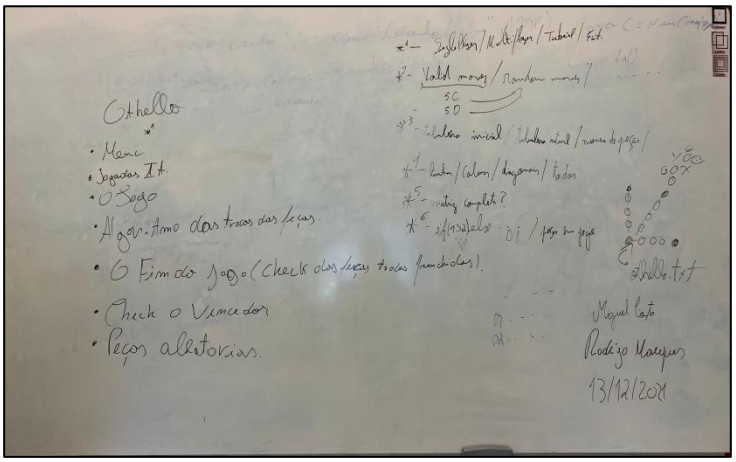

Figura 1:

Nesta primeira imagem podemos observar as ideias inicias que possuíamos e o protótipo da estrutura do nosso código.

Ideias como um menu, que acabou por ser descartado, as funções que possivelmente seriam necessárias para além das pedidas no enunciado, as mecânicas do jogo, etc...

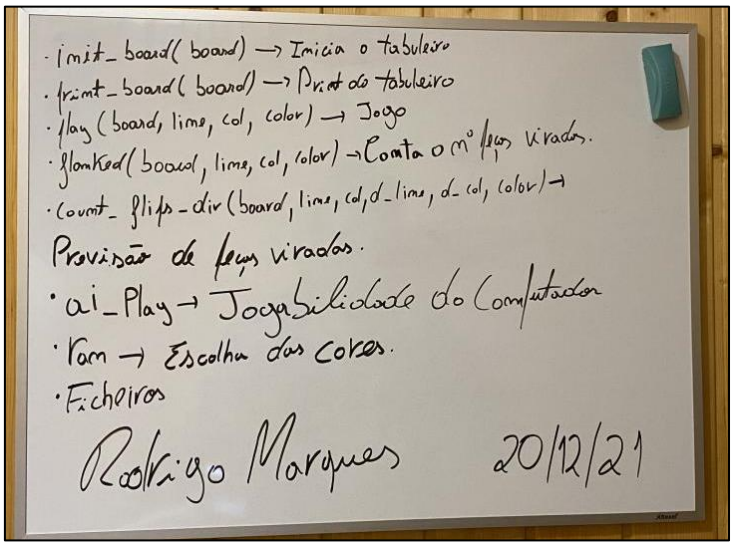

Figura 2:

Nesta foto encontram-se mais exemplos das funções e dos seus parâmetros que acabaram por fazer parte do jogo, tal como uma ideia mais desenvolvida da inteligência artificial que acabaria por avaliar todas os espaços onde é possível jogar e depois colocar a peça no espaço que vira mais peças (apesar desta estratégia não ser a mais otimizada o jogo todo).

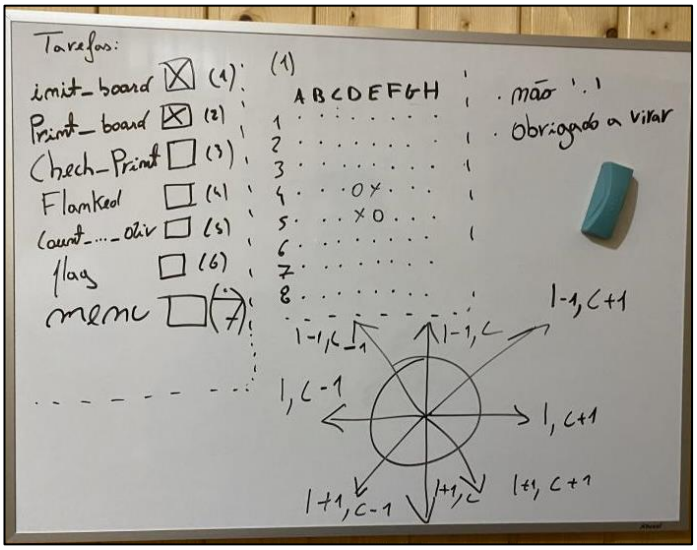

Figura 3:

Na fotografia ao lado está desenhado o tabuleiro do jogo no seu estado inicial, com o "X" a representar as peças pretas e o "O" a representar as peças brancas. Abaixo do tabuleiro encontra-se uma representação gráfica do método utilizado para verificar as peças tanto para a função count_flip_dir e na função flanked.

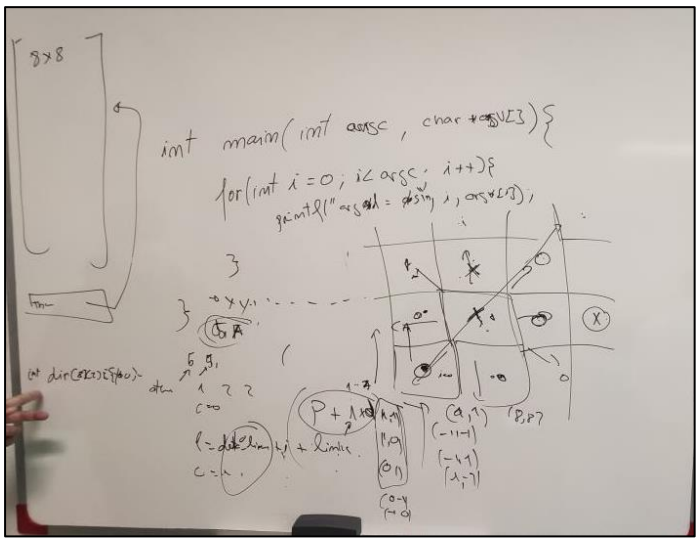

Figura 4:

Na última foto estão alguns apontamentos sobre a função anteriormente referida, tal como os valores que vão servir para procurar e as posições para onde procuraria. Para além disso também podemos ver os dois argumentos que o main terá de receber para ler o ficheiro jogadas.txt

O Código

Ao todo, foram escritas 17 funções que representam 382 linhas de código e, nesta parte do relatório, iremos descrever o funcionamento de cada uma das funções tal como erros e tentativas que realizamos ao longo do trabalho.

Em primeiro lugar as variáveis principais que são utilizadas como parâmetros em várias das funções:

f: f é o ponteiro que o programa utiliza para ler o ficheiro que é introduzido nos argumentos do main através do ficheiro jogadas.txt

board: A variável board é uma matriz de dimensão nove que irá representar o tabuleiro ao longo de todas as funções. Neste caso, foi necessário dar o tipo char a esta variável uma vez que as posições do vetor são representadas por caracteres e não por números.

color: A variável color representa a cor das peças do tabuleiro. Neste caso a cor preta seria 'X' e a cor branca seria 'O'. Assim sendo o tipo desta variável é char.

player: A variável player, de tipo char, é um vetor de dimensão dois que guarda os jogadores, ou seja, irá guardar de que cor são as peças do jogador e do computador.

check: check é uma variável de tipo inteiro que representa um vetor de dimensão 8 que guarda e controla os valores da direção que são verificadas por funções como a flanked.

score: A variável score é um vetor de tipo inteiro de dimensão dois que guarda a pontuação dos jogadores ao longo do jogo.

De seguida, iremos falar das funções que utilizam tais variáveis como parâmetros e que no seu todo, fazem o programa funcionar. Comecemos pelas funções mais simples, ou seja, as que não dependem de outras funções para funcionar:

init_board: A função init_board é uma função de tipo void, uma vez que não é necessário retornar nenhum valor e tem como parâmetros board e score. Esta função tem o propósito de iniciar o tabuleiro no seu estado inicial que é uma peça preta nas coordenadas 4E e 5D e as peças pretas nas coordenadas 4D e 5E. Se, ao ser iniciado, o programa não receber nenhum argumento, então estas serão as posições iniciais do jogo.

print_board: A função print_board é uma função de tipo void e recebe como parâmetro board. A função "desenha" o tabuleiro, no seu estado atual, no terminal que é atualizado ao longo do jogo.

flanked: A função flanked é do tipo inteiro uma vez que retorna valores inteiros. Esta função conta o número de peças viradas ao jogar numa certa linha, coluna ou diagonal. Se a jogada for válida retorna 1 e se a jogada for inválida retorna 0.

count_flip_dir: Esta função tem de tipo inteiro e recebe como parâmetros board, line, col, delta_line, delta_col e color. A função conta o número de peças que são viradas numa certa jogada. Retorna o número de peças viradas.

score_game: A função score_game tem de tipo void e recebe como color, score_player,score e player. O objetivo desta função é atualizar a variável score, ou seja, a pontuação dos jogadores para serem apresentadas no terminal enquanto o jogo decorre.

display_score: Esta função é de tipo void e recebe como parâmetros score e player. O objetivo deste código é "imprimir" o valor das pontuações dos jogadores anteiormente atualizadas pela função score_game.

choose: A função choose é de tipo void e recebe como parâmetros o ponteiro d_line, o ponteiro d_col e color. Tem como propósito colocar na consola o lugar onde o utilizar pode inserir a coordenada onde quer colocar a peça e guardá-las com a função scanf.

Ra: Esta função,de tipo char, recebe como parâmetros mini e maxi. Estes dois valores vão ser utilizados para determinar aleatóriamente quem começa o jogo, ou seja, quem fica com as peças pretas. Retorna a cor da peça, ou seja, o caracter que representa a peça.

save_player: Função de tipo void que recebe os parâmetros color e player. Esta função separa o jogador humano do computador. Caso o parâmetro color seja igual a 'X', o computador irá ficar com as peças pretas e o jogador as peças brancas. E vice-versa.

Agora a descrição das funções que dependem de outras funções para funcionar, assim dependendo de mais fatores:

verify: A função verify, de tipo inteiro, recebe como parâmetros board, color e check. O objetivo desta função é verificar se a jogada é válida ou não, utilizando a função flanked para determinar se a jogada é inválida. Se a jogada for válida retorna 1, se for inválida retorna 0.

play: Esta função, de tipo void, recebe como parâmetros board, line, col, color, score, player. Coloca uma peça do jogador na posição escolhida (line, col) e vira as peças do adversário.

wrong_play: A função wrong_play de tipo void, recebe os parâmetros board, color, check, score e player. O objetivo desta função é avisar o jogador que a jogada que inseriu é inválida, para isso, imprimindo no terminal uma mensagem.

next_move: Função de tipo void que recebe board, color, check, score e player como parâmetros. Esta função permite cumprir as etapas pretendidas para efetuar a troca de peças para do utilizador.

computer_play: Função de tipo void que tem como parâmetros board, color, check, score e player. Esta função permite ao computador fazer uma jogada e escolher a jogada que vira mais peças.

winner: Esta função é do tipo void e recebe como argumentos score e player. Com esta função mostra o tabuleiro final do jogo, a pontuação de cada jogador e quem foi o vencedor da partida.

load_game: A função load_game é de tipo char e recebe como parâmetros argc, argv, f, color, board, check, score e player. A função deste código é receber e ler um ficheiro que possuí coordenadas, que por sua vez, altera o estado inicial do tabuleiro.

game: Esta função de tipo void recebe os parâmetros board, check, color, score, player. Esta função executa o procedimento do jogo inteiro.

Para concluir esta parte do trabalho vamos também falar de iterações anteriores do nosso código, ideias que acabaram por ser descartadas e possíveis erros.

Anteriormente, na função flanked, possuíamos código que avaliava cada direção (as 8 direções anteriormente referidas) independentemente, o que resultava num código funcional, no entanto bastante mal otimizado pois com a atualização dessa parte do trabalho foi possível poupar cerca de 211 linhas de código. Também pensamos em implementar um menu, no entanto acabou por não ser necessário uma vez que o enunciado do trabalho apenas pede para fazer código de jogador contra computador. Anteriormente, como era explicitamente dito no enunciado, pensamos em não implementar um computador inteligente e que apenas selecionava uma posição válida no tabuleiro aleatoriamente, no entanto, como foi possível terminar o resto das funções a tempo, também adicionamos um fator de inteligência no computador.

Como possíveis erros, se o utilizador escrever alguma coisa fora do esperado (coordenadas como 5F, 3D, etc...) o programa irá parar de funcionar e retorna “Segmentation Fault”. O programa também não sabe associar as letras minúsculas às maiúsculas, ou seja, o programa consegue validar uma jogada como por exemplo, 3H, no entanto não consegue validar a jogada 3h, no entanto, o programa não retorna nenhum erro e simplesmente pede ao utilizador para inserir outra vez a coordenada o que acaba por não afetar a jogatina.

Conclusão

Após várias iterações e testes, foi possível implementar tudo o que foi pedido e tudo o que julgamos que fosse necessário para um bom funcionamento do programa, sem aparecerem erros em situações normais de jogo.